CISC 1600/1610 Computer Science I

Functions, continued

Professor Daniel Leeds
dleeds@fordham.edu
JMH 328A

What does this code do?

```
// funcA declaration

int main() {
  float a, b;
  cout << "Enter two numbers: ";
  cin >> a >> b;
  cout << funcA(a,b);
  return 0;
}

float funcA(float a, float b) {
  if(a>b)
    return a/b;
  else
    return b/a;
}
```

2

Specifications

Preconditions:
- What is assumed to be true when function is called

Postconditions:
- What will be true after the function is called (presuming preconditions are met)
  - What values are returned
  - What call-by-reference parameters are changed
  - What other output is produced

3

Example specification

- Include specs in comments of declaration

```
// funcA
// Precondition: takes two non-zero
// float inputs
// Postcondition: Function returns
// a float output such that output
// is result of dividing the bigger
// input by the smaller input
```

4

What does this code do?

```
// funcB declaration

int main() {
  int a, b;
  cout << "Enter two numbers: ";
  cin >> a >> b;
  cout << funcB(a,b);
  return 0;
}

??? funcB(int num1, int num2) {
  if(num1%num2==0)
    return "true";
  else
    return "false";
}
```

5

What does this code do?

```
int funcC(int a);

int main() {
  int a;
  cout << "Enter a number: ";
  cin >> a;
  cout << funcC(a);
  return 0;
}

int funcC(int a) {
  if(a==0)
    return a;
  else
    return a+funcC(a-1);
}
```

6

Recursion

When a function calls itself:

- Can be a simpler way to write a loop
- Can be used as a divide-and-conquer method

Recursive function design

Must have:

- Base case(s) – to eventually stop recursion
- Simplified recursive calls – each new call must bring us closer to reaching base case(s)

Complex problem, recursive solution

Towers of Hanoi:

- Start: all disks on peg 1 piled from big to small
- End: all disks on peg 3 piled from big to small
- Each step:
  - Move only one disk
  - Each disk can only be placed on top of a bigger disk

Recursive solution

Starting with 4 disks on peg 1:

- *Move top 3 disks from peg 1 to peg 2*
- Move remaining disk from peg 1 to peg 3
- *Move 3 disks from peg 2 to peg 3*

Function overloading

"Overloading" when multiple functions with same name but:

- different number of parameters
- different types of parameters

Compiler determines which function to use

Overloaded averaging function

```
float average(int num1, int num2) {
   return (num1+num2)/2.0;
}

float average(int num1, int num2, int
num3) {
   return ???;
}
```

Overloaded average function in action

```
int main()
{
  int numInputs; float in1, in2, in3;
  cout << "How many inputs?";
  cin >> numInputs;
  if(numInputs==2) {
    cout << "Give 2 numbers: ";
    cin >> in1 >> in2;
    cout << "Average: "
         << average(in1,in2) << endl;
  } else {
    cout << "Give 3 numbers: ";
    cin >> in1 >> in2 >> in3;
    cout << "Average: "
         << average(in1,in2,in3) << endl;
  }
  return 0;
}
```